





[Get support](#)

KB0623354

ServiceNow Secure Coding guide for Instance developers

 4367 Views  Last updated : Aug 12, 2024  Customer  Copy Permalink

Description

ServiceNow Secure Coding Guide

Table of Contents

1. Overview
2. Generic Secure Coding Recommendations
 - 2.1 Input Sanitization
 - 2.2 File Operations
 - 2.3 Encryption and sensitive information
 - 2.4 Secure Communication
 - 2.5 Authentication and authorization
3. Input & Output sanitization
 - 3.1 escapeScript
 - 3.2 escapeHTML
 - 3.3 unescapeHTML
 - 3.4 getHTMLValue
 - 3.5 cleanURL
 - 3.6 enforceRelativeURL
 - 3.7 removeInvalidChars
 - 3.8 getNumeric
 - 3.9 escapeAllQuotes
 - 3.10 escapeTicks
 - 3.11 newLinesToBreaks
 - 3.12 normalizeWhitespace
 - 3.13 escapeNonPrintable
 - 3.14 dotToUnderBar
 - 3.15 escapeQueryTermSeparator
 - 3.16 escapeForHomePage
4. Data validation
 - 4.1 validate
 - 4.2 parse (GlideXMLDocument)

- 4.3** parse (GlideXMLUtil)
 - 4.4** validateXML
 - 4.5** areEqual
 - 4.6** isURLWhiteListed
 - 4.7** isBase64
 - 4.8** isEligibleSysID
- 5.** Session Management
 - 5.1** get
 - 5.2** disconnect
 - 5.3** getSessionKey
 - 5.4** getUserName
 - 5.5** getUID
 - 5.6** getUser
 - 5.7** isLoggedIn
 - 5.8** getSessionID
 - 5.9** hasRole
 - 5.10** getRoles
 - 5.11** getSessionToken
 - 5.12** getHttpSession
 - 5.13** getSession
- 6.** Secure Communication
 - 6.1** setSignerKey
 - 6.2** setSignerCertificate
 - 6.3** getSignedDocument
 - 6.4** wantSigned
 - 6.5** setSignatureKeys
 - 6.6** SOAPSigner
 - 6.7** sign
- 7.** Secure Data Access
 - 7.1** GlideRecord
 - 7.2** GlideRecordSecure
- 8.** Logging
 - 8.1** Logging
- 9.** Encryption and Hashing
 - 9.1** GlideEncrypter
 - 9.2** encrypt
 - 9.3** decrypt
 - 9.4** generateKey
 - 9.5** generate
 - 9.6** generateRand
 - 9.7** generateSecureRand
 - 9.8** getRandomString

Resolution

1. Overview

This article provides an overview of application security-related GlideScriptable classes and methods offered by ServiceNow, to assist and educate developers while creating and modifying the code on the target Instance.

This article is organized by security topics and respective attack vectors and is highlighted with both examples and security/functional impact. This provides the programmer with an understanding of the security risks, and how to remediate them.



Note: Apply all of the recommendations in this article to a non-production instance before making the changes to the production instance.

2. Generic Secure Coding Recommendations

This section talks about generic Java secure coding recommendations suggested by Industry experts.

2.1 Input Sanitization

It is important to ensure proper Input sanitization of all data received from any external source before processing it via the instance, in order to avoid any kind of malicious data injection issues

SQL Injection

It is important to use proper data sanitization and validation with the help of escaping both simple and double quotes or any other character that could lead to SQL Injection, it is also strongly recommended to use prepared statements when dealing with dynamic SQL queries.

Stored and reflected XSS

User input is untrusted, it is important to sanitize input against any kind of javascript injection at both input and output during rendering time. This will prevent from both reflected and stored XSS issues

CSRF

It is important to ensure the authenticity of the source of any request before performing updates on the system, especially when dealing with sensitive information. Different mechanisms can be used to implement proper mitigation against CSRF attacks

Command injection

Sanitize input data from any sensitive or invalid characters including null bytes, non-printable characters, and any other OS injection special characters such as "|", ";", etc ...

2.2 File Operations

File manipulation is a critical process, it is important to ensure that any file operation is handled properly in order to avoid illegal file system access or even remote code execution

This could be done by performing strict file names, paths, and content validation before processing any input file from the user and use safe storage mechanisms on the backend

File/Path Sanitation

It is important to ensure that file names and paths are properly sanitized when received from the user in order to avoid any directory traversal related security issues, this can be done by sanitizing all file names and paths against invalid characters or any file/path separator allowed by the os.

Content Validation

It is important to validate the file content and ensure its consistency before processing it, this could be done by specific parsers or by validating the mime type against the file signature.

Special handling should be used when dealing with compressed files in order to avoid any issues when extracting the packages before processing it..

File Storage and Extension

It is important to ensure safe storage for the uploaded files and downloads to restrict access to only limited and safe file extensions without processing any server-side code.

It is also important to limit and enforce allowed file extensions and perform strict validation in order to avoid any upload or download of malicious file types.

2.3 Encryption and sensitive information

It is important to handle carefully any sensitive information in order to ensure proper privacy of any information including passwords, encryption keys, credit card numbers, etc. or any other proprietary and critical data

Storing Sensitive Information

Never store sensitive information such as user credentials or encryption keys in clear text on the system, use strong hashing functions or encryption mechanisms instead

Hardcoded Information

Do not hardcode any credentials or other critical information such as encryption keys and passwords within the scripts.

Logging

Do not log any sensitive information on the instance, use wildcard (*) to mask the information if any logging is needed

2.4 Secure Communication

It is important to ensure a safe communication for any sensitive information in order to avoid any information theft during transfers

Use of Encrypted Communication

Always use encrypted communication channels and protocols when transferring any sensitive information, it is important to use strong and industry-standard protocols and not rely on custom encryption on to of clear-text protocols

POST vs GET

Do not send any sensitive information such as session IDs or passwords in URLs using GET method, use the POST method instead.

2.5 Authentication and authorization

Ensure that strict authentication and authorization checks are enforced in order to apply the least privileges access standards.

Apply role-based access lists to ensure that only users with appropriate roles can access limited data according to the business needs.

3. Input & Output Sanitization

User-supplied data either via User-agent or through programmatic invocation should be treated as untrusted data, and should be sanitized appropriately before processing. Failure to properly sanitize input or output may introduce major security risks.

This section discusses various GlideScriptable classes and methods for input and output sanitation, for use in customer scripts. It also provides some ServiceNow examples.

3.1 escapeScript

SCOPE: This method is NOT accessible in the script running in a package scope.

The escape script method uses output replacement patterns to prevent Javascript injection that might potentially lead to Cross-site scripting. It is important to escape all user input to avoid any security exposure.

Method definition

Class: GlideSecurityUtils

Definition: String escapeScript(String s)

Security impact

If the user input is not escaped, this may allow a malicious user to inject javascript tags with HTML content when rendered, which may lead to javascript code injection (XSS)

Functional impact

This should not have any impact when the user input is included within the HTML output page since the escape function will convert javascript tags into their HTML notation, which will render properly. When used for other purposes, it is important to note that tags characters will be converted to their HTML encoded equivalent such as: "<" to "<".

This may cause some troubles if user input is meant to be evaluated or some while performing hashing comparison. On the other side if the system escape property is already set this may lead to rendering the content incorrectly due to double escaping.

Example

```
var mydata = "<script> alert(1)</script>"
var escape = new GlideSecurityUtils()
mydata = escape.escapeScript(mydata)
gs.print(mydata)
```

Result:

```
Script: &lt;script&gt; alert(1)&lt;/script&gt;
```

3.2 escapeHTML

SCOPE: This method is NOT accessible in the script running in a package scope.

The escaped HTML method can be used to escape any illegal characters that might cause the UI to render improperly or trigger any client-side attacks such as Javascript or HTML injection. It is important to escape all user input to avoid any security exposure.

Method definition

Class: GlideStringUtil

Definition: String escapeHTML(String s)

Security impact

If the user input is not escaped, this may allow a malicious user to inject javascript code within HTML content using different characters such as closing quote and double quote. For example, appending malicious JS code when the page is rendered may lead to javascript code injection (XSS)

Functional impact

When the user input is used to be included within the HTML output page, this should not have any impact since the escape function will convert javascript tags into their HTML notation that will render properly. In some cases when used for different purposes it is important to note that tags characters will be converted to their HTML encoded equivalent such as: "<" to "<,"

Note: This may cause issues if user input is meant to be evaluated or perform any hashing comparison. On the other side, if the system escape property is already set, this may lead to rendering the content incorrectly due to double escaping.

Example

```
var mydata = '"<>&'
var escape = new GlideStringUtil()
mydata = escape.escapeHTML(mydata)
gs.print(mydata)
```

Result:

```
*** Script: &quot;&lt;&gt;&amp;
```

3.3 unescapeHTML

SCOPE: This method is NOT accessible in a script running in a package scope.

This method uses a matching table and un-escape all HTML characters to their original value and performs the following substitutions:

- < to <
- > to >
- to " " (space)
- & to &
- " to " (double quote)

This may be used to unescape HTML tags and restore previous user input to its original value by converting back HTML encoded characters to their initial values. For any reasoning where the developer would need to allow the user input as is, this method definition can be used, however please understand that unescaping any data from untrusted source will expose your instance to potential Security risk.

Method definition

Class: GlideStringUtil

Definition: String unescapeHTML(String s)

Security impact

Since the hardening guide provides properties to perform user sanitization, it is important to not render user content issue from a call to `unescapeHTML`. With malicious user data, this may lead to XSS attacks.

Functional impact

N/A

Example

```
var mydata = '"&lt;&gt;&amp;';
var escape = new GlideStringUtil()
var unescaped = escape.unEscapeHTML(mydata)
gs.print(unescaped)
```

Result:

```
*** Script: "<>&
```

3.4 getHTMLValue

The `getHTMLValue` converts any illegal HTML input value to its HTML notation, thus encoding value of illegal characters passed as an argument. This will prevent any client side injection while rendering user supplied input onto the browser. This may be used to clean any HTML illegal value when receiving user input to avoid any HTML and JS injections.

Method definition

Class: `GlideStringUtil`

Definition: `String getHTMLValue(String s)`

Security impact

If the user input is rendered as received, this may allow a malicious user to inject HTML or javascript code within the rendered page using different characters such as closing quote and double quote for example to append malicious JS code which may lead to javascript code injection (XSS)

Functional impact

When the user input is used to be included within HTML output page, this should not have any impact since the escape function will convert javascript tags into their HTML notation that will render properly. When used for other purposes it is important to note that tags characters will be converted to their HTML encoded equivalent such as: `"<"` to `"<"`

This may cause issues if user input is meant to be evaluated or while performing hashing comparison. On the other side if the system escape property is already set this may lead to render the content incorrectly due to double escaping.

Example

```
var mydata = '&'
var escape = new GlideStringUtil()
var htmlvalue = escape.getHTMLValue(mydata)
gs.print(htmlvalue)
```

Result:

```
*** Script: &amp;
```

3.5 cleanURL

The cleanURL method is specially designed to clean URLs by removing any suspicious encoding and processing different protocols such as "data:", javascript, vbscript:, etc., to prevent any reflected or DOM based Cross site scripting attacks.

The method will remove any URL that matches specific patterns based on the regular expressions defined to detect different browser injections within a URI.

Method definition

Class: GlideSecurityUtils

Definition: String cleanURL(String s)

Security impact

It is important to clean any URI scheme before processing it or passing it within a redirection, or else this may lead to javascript code execution on the client side or malicious redirection.

Functional impact

This method will clean and delete any URI scheme containing javascript code injection. This should not have a functional impact unless specific URI scheme are meant to be used within the system, the javascript code will be stripped.

Example

```
var myurl = 'javascript%3Aalert(1)'
var myCleaner = new GlideSecurityUtils()
var clean = myCleaner.cleanURL(myurl)
gs.print(clean)
```

Result:

```
*** Script: null
```

3.6 enforceRelativeURL

SCOPE: This method is NOT accessible in script running in a package scope.

The enforceRelativeURL method offers the possibility to strip down a full URL and returns only the page name to be used as a relative location.

This is useful to sanitize user inputs and avoid other user-input injections such as remote file includes, unsafe redirects etc.

Use this method to enforce relative urls in case of redirection on it's input to avoid any kind of malicious user redirect from the instance.

Method definition

Class: GlideSecurityUtils

Definition: String enforceRelativeURL (String url)

Security impact

During a redirection, when the target URL is based on the user input (Example: taken from GET parameter) this may be abused by a malicious user to redirect a legitimate user to a malicious website. It is important to enforce the redirection to relative URLs so the user will be redirected to one of the instance pages instead of being redirected to a remote untrusted website.

Functional impact

This method should not have any functional impact unless it is meant to redirect a user to remote websites during specific operations (such as logout). In that case it is recommended to use the URL redirection whitelist system property to prevent any abuse of URL redirection.

Example

```
var myurl='http://evildomain.com/test.do'  
var myCleaner=new GlideSecurityUtils()  
var relativeURL=myCleaner.enforceRelativeURL(myurl)  
gs.print(relativeURL)
```

Result:

```
*** Script: test.do
```

3.7 removeInvalidChars

SCOPE: This method is NOT accessible in script running in a package scope.

The removeInvalidChars method offers the developer to strip off invalid and special characters that may be harmful within the XML content.

This method will receive a string as an input and will provide an output by removing all XML invalid characters defined within the following ranges:

- 0 .. 8
- 11, 12
- 14 - 31 (inclusive)
- 55296 - 57343 (inclusive)
- 65534 - 65535 (inclusive)

Use this method on an XML user input to sanitize the data and remove any invalid characters that may lead to XML content corruption that can be used for XML injection or other corruption attacks.

Method definition

Class: GlideXMLUtil

Definition: String removeInvalidChars (String s)

Security impact

If not properly sanitized, XML input may be corrupted with invalid characters that may lead to a wrongful interpretation by the backend parser. This may lead to potential XML injection and corruption bugs.

Functional impact

Be sure that the service does not need any invalid characters. Even if the stripped characters are generally unprintable it is important to review the list and ensure that none of these characters are needed for any functional purpose.

Example

N/A

3.8 getNumeric

SCOPE: This method is NOT accessible in script running in a package scope.

The getNumeric method takes a string as an input and extracts only numeric characters. This is a useful routine to sanitize user input when the script is expecting only numeric values to be processed.

Use this method when expecting only numeric values from the user input, since it will clean all user input and keeps only numeric characters (0-9).

Method definition

Class: GlideStringUtil

Definition: String getNumeric (String str)

Security impact

If not properly sanitized a user may input alphabetic characters where the application is expecting a numeric value that may lead to a type confusion, that would trigger validation bugs or Information disclosure.

Functional impact

This method strips all non-numeric characters so user input will be different from the output that may lead to some functional issues (as an example processing user password). It is important to be sure that the application that is being developed only needs numeric values to avoid any functional issues.

Example

```
var mystring = '123 test 456 String 789 cleaning'  
var myCleaner = new GlideStringUtil()  
var onlyNumeric = myCleaner.getNumeric(mystring)  
gs.print(onlyNumeric)
```

Result:

```
*** Script: 123456789
```

3.9 escapeAllQuotes

SCOPE: This method is NOT accessible in script running in a package scope.

Use this method to escape all quotes (single and double) within a given string, as an input by stripping them off to prevent malicious injections.

Method definition

Class: GlideStringUtil

Definition: String escapeAllQuotes (String source)

Security impact

This applies when user input is being used to construct dynamic string content such as SQL queries, HTML input, commands parameters etc. In those cases single and double quotes may be used to abuse the application to perform different attacks such as SQL or command injection.

Functional impact

This function may affect the rendering page where the single and double quotes are removed and impact the user reading experience.

Example

```
var mystring = "let's escape some quotes"
var myCleaner = new GlideStringUtil()
var escapeQuote = myCleaner.escapeAllQuotes(mystring)
gs.print(escapeQuote)
```

Result:

```
*** Script: lets escape some quotes
```

3.10 escapeTicks

SCOPE: This method is NOT accessible in script running in a package scope.

The escapeTicks may be used as an escape function for strings. However, instead of stripping down the characters as per escapeAllQuotes, it generates an escaped output using backslash "\".

Method definition

Class: GlideStringUtil

Definition: String escapeTicks (String source)

Security impact

This generally applies when user input is used to construct dynamic string content such as SQL queries, HTML input, commands parameters etc. In those cases single and double quotes may be used to abuse the application to perform attacks such as SQL or command injection..

Functional impact

This function may affect the rendering page where the single and double quotes are escaped and impact the user reading experience.

Example

```
var mystring = "let's try escapeTicks"
var myCleaner = new GlideStringUtil()
var escaped = myCleaner.escapeTicks(mystring)
gs.print(escaped)
```

Result:

```
*** Script: let\'s try escapeTicks
```

3.11 newLinesToBreaks

SCOPE: This method is NOT accessible in script running in a package scope.

This method replaces the new line character "\n" to an HTML break code "
" to sanitize the user input into a clean HTML code.

Use this method against user input where new lines are expected to be rendered on the page, thus sanitizing any malformed input with "\n" as input to a valid HTML break tag.

Method definition

Class: GlideStringUtil

Definition: String newLinesToBreaks (String value)

Security impact

TBD

Functional impact

Use this function to ensure a proper rendering of linefeed received as a raw or file input from the user converting linefeed to HTML break to ensure a good user experience.

Example

```
var mystring = "new line break \n, this is after the break"
var myCleaner = new GlideStringUtil()
var replaceNewLine = myCleaner.newLinesToBreaks(mystring)
gs.print(replaceNewLine)
```

Result:

```
*** Script: new line break <br/>, this is after the break
```

3.12 normalizeWhitespace

SCOPE: This method is NOT accessible in script running in a package scope.

This method Normalizes the whitespace in the given string by transforming all carriage returns, line feeds and tabs to spaces, and then all leading, trailing, and duplicate spaces are removed.

Use this method against user input where new lines are expected to be rendered on the page, thus sanitizing any malformed input with "\n" as input to a whitespace.

Method definition

Class: GlideStringUtil

Definition: String normalizeWhitespace (String value)

Security impact

Misinterpreted carriage returns may be used as a different type of injection. User input may define a new line which will trick the application to consider a simple input as two or multiple lines of input leading to unexpected application behavior. By converting all carriage returns to a white space we avoid such kind of vulnerability by forcing the application to use the whole input as a single line.

Functional impact

This function may affect user experience if the input data is used to be rendered on the output page. In that case, the newLinesToBreaks function is more suitable.

Example

```
var mystring = "test with \n (new line) and \t (tabulation)"
var myCleaner = new GlideStringUtil()
var normalizedString = myCleaner.normalizeWhitespace(mystring)
gs.print(normalizedString)
```

Result:

```
*** Script: test with (new line) and (tabulation)
```

3.13 escapeNonPrintable

SCOPE: This method is NOT accessible in script running in a package scope.

This method escapes all non-printable characters by converting them to a printable notation.

Use this method against user input where non-printable characters may be dangerous and may lead to undefined behavior of the application.

Method definition

Class: GlideStringUtil

Definition: String escapeNonPrintable (String value)

Security impact

Unprintable characters may be misinterpreted by the application and to different kind of injection (Examples: Null bytes injection, logs poisoning with backspace etc.). Cleaning and escaping such characters is highly recommended before processing them.

Functional impact

It is always safe to convert unprintable characters to their hexadecimal notation, or escape them during render process to ensure a better user experience. This may affect the user experience in some places where user code page is not Latin, and may require different interpretation of those data or codes.

Example

```
var mystring = "test \x09 non \x00 printable \x07 chars"
var myCleaner = new GlideStringUtil()
var escapedString = myCleaner.escapeNonPrintable(mystring)
gs.print(escapedString)
```

Result:

```
*** Script: test \t non \u0000 printable \u0007 chars
```

3.14 dotToUnderBar

SCOPE: This method is NOT accessible in script running in a package scope.

This method will replace all dots "." with an underbar "_" character

This method is generally used against user input where the application expects a filename or path when either trying to access a file on the file system or while constructing a file path.

Method definition

Class: GlideStringUtil

Definition: String dotToUnderBar (String value)

Security impact

Filenames and paths should be carefully handled and properly sanitized when received from a user, a failure to perform a proper sanitation a malicious user may inject a path traversal pattern "../" to bypass current restrictions and access files and folders on the file system and this leading to the exploitation of a directory/path traversal vulnerability.

This method will convert all dots to underbar which will help to mitigate direct directory/path traversal, however additional precautions and sanitation should be taken in consideration when constructing filenames and paths from user input.

Functional impact

This should not have a functional impact, however it is important to ensure that the filenames stored on the database table have the same name as the ones on the file system to keep consistency.

Example

```
var filename = "../../../../etc/passwd"
var myCleaner = new GlideStringUtil()
var cleanFilename = myCleaner.dotToUnderBar(filename)
gs.print(cleanFilename)
```

Result:

```
*** Script: __/__/__/__/__/etc/passwd
```

3.15 escapeQueryTermSeparator

SCOPE: This method is NOT accessible in script running in a package scope.

This method will replace all query term separator "^" by appending another "^" to avoid any SQL query misinterpretation when the character is used as a string instead of a query term separator.

This method is generally used against user input where the application expects string to construct an SQL query.

Method definition

Class: GlideStringUtil

Definition: String escapeQueryTermSeparator (String value)

Security impact

Make sure you perform input sanitation against user input before using it to construct any SQL query. A wrongly placed query separator "^" may lead to a different SQL query against the database that may be abused by a malicious user. With the help of this method, an escaped separator will be interpreted as a string and not used to construct SQL queries terms.

Functional impact

This should not have a functional impact.

Example

```
var myquery = "test^Test"
var escaper = new GlideStringUtil()
var escapedQuery = escaper.escapeQueryTermSeparator(myquery)
gs.print(escapedQuery)
```

Result:

```
*** Script: test^^Test
```

3.16 escapeForHomePage

SCOPE: This method is NOT accessible in script running in a package scope.

This method is similar to escapeHTML but used by the homepage rendering code which requires more aggressive escaping.

Method definition

Class: GlideStringUtil

Definition: String escapeForHomePage (String source)

Security impact

NA

Functional impact

NA

Example

```
var mystring = "<test> string \n to escape"
var escaper = new GlideStringUtil()
var escapedString = escaper.escapeForHomePage(mystring)
gs.print(escapedString)
```

Result:

```
*** Script: %3ctest%3e string \n to escape
```

4. Data validation

It is important to perform proper data validation before processing it, since passing unexpected data types may drive the application into an unexpected behavior. In this section we will go through various GlideScriptable classes and methods related to data validation that may be used within customer scripts to validate the correctness of the input data, this will be illustrated by ServiceNow examples.

4.1 validate

SCOPE: This method is NOT accessible in script running in a package scope.

This method will take a string as input and return True/False if the string represents a valid 32 bits IP address or not.

Method definition

Class: SncAddress32Bit

Definition: boolean validate(String pAddress)

Security impact

Failing to perform data validation may lead to unexpected application behavior and create serious security risks. Ensure we receive a well formatted input as the expected format before processing it.

Functional impact

This should not have a functional impact.

Example

```
var myIP = "192.168.398.1"
var validator = new SncAddress32Bit("127.0.0.1")
var isvalid = validator.validate(myIP)
gs.print(isvalid)
```

Result:

```
*** Script: false
```

4.2 parse (GlideXMLDocument)

This method will take an XML string as input and return True/False if the string represents a valid XML document or not.

Method definition

Class: GlideXMLDocument

Definition: boolean parse(String xml)

Security impact

Failing to perform data validation may lead to unexpected application behavior and may lead to serious security risks. Ensure we receive a well formatted XML input as the expected format before processing it to avoid any XML injection or poisoning when parsing it.

Functional impact

This should not have a functional impact.

Example

```
var myXML="<xml><test>test Tag</wrongTAG></xml>"
var validator=new GlideXMLDocument()
var isvalid=validator.parse(myXML)
gs.print(isvalid)
```

Result:

```
org.xml.sax.SAXParseException; lineNumber: 1; columnNumber: 22; The element type
"test" must be terminated by the matching end-tag "</test>".
*** Script: false
```

4.3 parse (GlideXMLUtil)

SCOPE: This method is NOT accessible in script running in a package scope.

This method will take an XML string as input and return True/False if the string represents a valid XML document or not.

Method definition

Class: GlideXMLUtil

Definition: boolean parse(String xml)

Security impact

Failing to perform data validation may lead to unexpected application behavior and may lead to serious security risks. It is important to ensure that we receive a well formatted XML input as the expected format before processing it to avoid any XML injection or poisoning when parsing it.

Functional impact

This should not have a functional impact.

Example

```
var myXML = "<xml><test>test Tag</wrongTAG></xml>"
var validator = new GlideXMLUtil()
var isvalid = validator.parse(myXML)
gs.print(isvalid)
```

Result:

```
org.xml.sax.SAXParseException; lineNumber: 1; columnNumber: 22; The element type
"test" must be terminated by the matching end-tag "</test>".
*** Script: null
```

4.4 validateXML

SCOPE: This method is NOT accessible in script running in a package scope.

This method will take an XML string as an input and return True/False, depending on if the string represents a valid XML document or not. This method returns a parsing error message if the XML does not validate or otherwise returns null. forgiveUnenclosed causes the XML to be wrapped in <xml></xml> tags

This method is generally used to perform data validation against XML input by the user to validate if the input is a valid XML document or not.

Method definition

Class: GlideXMLUtil

Definition: String validateXML(String xml, boolean nsAware, boolean forgiveUnenclosed)

Security impact

Failing to perform data validation may lead to unexpected application behavior and may lead to serious security risks. Ensure we receive a well formatted XML input as the expected format before processing it to avoid any XML injection or poisoning when parsing it.

Functional impact

This should not have a functional impact.

Example

```
var myXML = "<xml><test>test Tag</wrongTAG></xml>"
var validator = new GlideXMLUtil()
var isvalid = validator.validateXML(myXML,true,true)
gs.print(isvalid)
```

Result:

```
*** Script: Error at line (1) The element type "test" must be terminated by the
matching end-tag "</test>".
```

4.5 areEqual

SCOPE: This method is NOT accessible in script running in a package scope.

This method will take two objects as input and validate if they are equal or not.

This method is used to compare proper objects, to check if the input object is equal to another one.

Method definition

Class: GlideObjectUtil

Definition: boolean areEqual(Object a, Object b)

Security impact

Failing to perform secure comparison between objects may lead to an unexpected application behavior and may include serious security risks. Ensure a proper and secure objects comparison, especially when dealing with users rights and authentication/authorization mechanisms.

Functional impact

This should not have a functional impact.

Example

N/A

4.6 isURLWhiteListed

SCOPE: This method is NOT accessible in script running in a package scope.

This method will check the URL (received as input) against the system defined whitelisted URLs and returns either True when found in the Whitelist property.

This method is used to validate the user input URL and ensure that this URL is within the URL whitelist entries before processing it or performing any redirects.

Method definition

Class: GlideSecurityUtils

Definition: boolean isURLWhiteListed(String urlStr)

Security impact

Even with the existence of system property, it is best practice to enforce a check of whitelisted URLs during specific processes (such as logout redirect). It is important to validate before performing any custom user redirect to avoid any redirection to a malicious URL provided by an attacker.

Functional impact

This should not have a functional impact.

Example

```
var myURL = "http://evil.com/badscrip.do"
var validator = new GlideSecurityUtils()
var isWhitelisted = validator.isURLWhiteListed(myURL)
gs.print(isWhitelisted)
```

Result:

```
*** Script: false
```

4.7 isBase64

SCOPE: This method is NOT accessible in script running in a package scope.

This method will check the given input against a matcher and return True if the input is a valid base64 encoded value.

This method is used to check the user input string and ensure that this represents a valid base64 string before processing it (decoding).

Method definition

Class: GlideStringUtil

Definition: boolean isBase64(String test)

Security impact

Trying to base64 decode a string that is not a valid base64 input may lead to unexpected behavior and generally throws exceptions. It may also cause information disclosure. Validate the format of the input string before passing it to a base64 decoding routine.

Functional impact

This should not have a functional impact.

Example

```
//(adding a "*" to corrupt the base64 format)
var base64="GethdTYehdtshetB*"
var validator=new GlideStringUtil()
var isValid=validator.isBase64(base64)
gs.print(isValid)
```

Result:

```
*** Script: false
```

4.8 isEligibleSysID

SCOPE: This method is NOT accessible in script running in a package scope.

This method will check if the given input is well formatted as a valid sys_id and returns True if the input matches ServiceNow standards.

Use this method to check the user input string and ensure that the data represents a valid sys_id before using it to reference objects on the database to keep code consistency. ServiceNow standards for a correct sys_id is defined as sequence of 32 valid hexadecimal representation where all the characters should be within the range [0-9a-fA-F].

Method definition

Class: GlideStringUtil

Definition: boolean isEligibleSysID(String id)

Security impact

Failing to validate any input used within the database may lead to unexpected behavior and serious security risks when building and executing SQL queries. Validate any user input before using it as a sys_id within the database.

Functional impact

This should not have a functional impact.

Example

```
var sysID = "62826bf03710200044e0bfc8bcbe5df1"
var validator = new GlideStringUtil()
var isElig = validator.isEligibleSysID(sysID)
gs.print(isElig)
```

Result:

```
*** Script: true
```

5. Session Management

Proper session management is crucial from Security standpoint. Ensure the user is authenticated with correct session associated to its roles before exposing any information from the instance. In this section we will discuss GlideScriptable classes and session management methods that may be used within customer scripts. We will also illustrate it with ServiceNow examples.

5.1 get

SCOPE: This method is NOT accessible in script running in a package scope.

This method will fetch and return the current session. If the session is not created, it will create new user session and return the value.

Use this method if there is a need to fetch or verify current user session. For example, To get the sys_ids of all currently logged in users

Method definition

Class: GlideSession

Definition: GlideSession get()

Security impact

Validate the current user session and ensure that the session is valid.

Functional impact

This should not have a functional impact.

Example

The following script fetches the list of currently logged in users:

```
var loggedIn = [];  
var sessions = GlideSessions.get().getLoggedInSessionList();  
var it = sessions.iterator();  
  
while (it.hasNext()) {  
    var session = it.next();  
    var id = session.getUser();  
    var user = new GlideRecord('sys_user');  
    user.get('user_name', id);  
    loggedIn.push(user.sys_id);  
}  
gs.log(loggedIn);
```

5.2 disconnect

SCOPE: This method is NOT accessible in script running in a package scope.

This method will disconnect the current user session. Any active database connections for the session are released, as are any held Mutexes

This method may be used to end/kill the current session.

Method definition

Class: GlideSession

Definition: void disconnect()

Security impact

It is important to perform a clean disconnection of the user and kill the session when no more needed.

Functional impact

This should not have a functional impact.

Example

```
GlideSession.disconnect();
```

Or

```
GlideSession.disconnect(true);
```

Result:

The current session is ended/killed

5.3 getSessionKey

SCOPE: This method is NOT accessible in script running in a package scope.

Returns the unique session key for current session object. **Note:** session ID is not unique

Method definition

Class: GlideSession

Definition: String getSessionKey()

Security impact

N/A

Functional impact

This should not have a functional impact.

Example

```
gs.print(gs.getSession().getSessionKey())
```

Result:

```
*** Script: A75587E2589F72B015A74D76303BB877
```

5.4 getUsername

Returns the default username (login name) currently logged in to the application. (for example, jsmith).

Method definition

Class: GlideSession

Definition: String getUsername()

Security impact

N/A

Functional impact

This should not have a functional impact.

Example

```
// the following example adds a comment when closing a task  
current.comments = "Closed by " + gs.getUsername() + " at " + gs.nowDateTime();  
gs.addInfoMessage("Close comment added");
```

5.5 getUID

SCOPE: This method is NOT accessible in script running in a package scope.

Returns the sys_id of current user with current session, otherwise returns undefined.

Method definition

Class: GlideSession

Definition: String getUID()

Security impact

N/A

Functional impact

This should not have a functional impact.

Example

For example, create an event queue based on comments that logs sys_id and username of the user.

```
if (current.operation() != 'insert' && current.comments.changes()) {  
    gs.eventQueue("incident.commented", current, gs.getUID(), gs.getUserName());  
}
```

5.6 getUser

This method may be used in a business rule or other server-side javascript to retrieve user object of a currently logged in user.

A use case may be if you want to validate or retrieve the current user role before allowing any server side operation. Once you get the current user object, you may use that with other methods as mentioned below:

- myUserObject.getRoles() -- returns all the roles of the current user
- myUserObject.isMemberOf('Group'); -- returns true or false if the current user is a member of the provided group (takes a group name or sys_id as its argument). isMemberOf() returns false if the group is inactive, even if the user is a member
- myUserObject.hasRole() -- returns true or false if the current user has the provided role (takes a role name as its argument)

Method definition

Class: GlideSession

Definition: User getUser()

Security impact

N/A

Functional impact

This should not have a functional impact.

Example

```
var myUserObject = gs.getUser()  
gs.print(myUserObject)
```

Result:

```
*** Script: com.glide.sys.User@b2e401
```

5.7 isLoggedIn

Determines if the input user is currently logged in. Returns True if the current user is logged in.

Use this method to check if any user is logged in, and determine proper actions to taken for authenticated vs. unauthenticated users. It may be used as a part of any server side script or business rule when you need to restrict user action, if unauthenticated.

Method definition

Class: GlideSession

Definition: boolean isLoggedIn()

Security impact

Limit access to the instance data to only authenticated users. Unauthenticated users should only have access to minimal required data for functional reasons.

Functional impact

This should not have a functional impact.

Example

```
If (gs.isLoggedIn()) {  
    // authenticated_action  
} else {  
    // unauthenticated_action  
}
```

5.8 getSessionID

Retrieves the GlideSession session ID.

This method can be used to terminate current user session.

Method definition

Class: GlideSession

Definition: String getSessionID()

Security impact

N/A

Functional impact

This should not have a functional impact.

Example

```
var s = gs.getSessionID();  
GlideTransactionManager.kill(s);
```

5.9 hasRole

This method checks if the user has a role supplied as an argument. Returns True if the current user has the provided role (takes a role name as its argument).

Can be sued with any server side script to validate the roles of the user (like Script Include or processor).

Method definition

Class: GlideSession

Definition: boolean hasRole(String appRoles)

Security impact

Check if the user requesting such action is actually authorized before performing any of CREATE READ UPDATE DELETE operation on the table records.

Functional impact

This should not have a functional impact.

Example

```
var myUserObject = gs.getUser()  
gs.print(myUserObject.hasRole('admin'))
```

5.10 getRoles

Returns all the roles of the current user. A list of comma separated role, including the default roles. The list of roles does not reflect any changes made during the current user session. To get the updated list of roles, the user must log out and log back in.

Can be used with any server side script to validate the roles of the user (like Script Include or processor)

Method definition

Class: GlideSession

Definition: String getRoles()

Security impact

Check if the user requesting such action is actually authorized before performing any of CREATE READ UPDATE DELETE operation on the table records.

Functional impact

This should not have a functional impact.

Example

```
var myUserObject = gs.getUser()  
gs.print(myUserObject.getRoles());
```

Instances using the Contextual Security Manager must use `gs.getSession().getRoles()` .

```
gs.print(gs.getSession().getRoles());
```

5.11 getSessionToken

Returns the GlideSession token.

Method definition

Class: GlideSession

Definition: String getSessionToken()

Security impact

N/A

Functional impact

This should not have a functional impact.

Example

```
gs.getSessionToken();
```

5.12 getHttpSession

SCOPE: This method is NOT accessible in script running in a package scope.

Retrieves a reference to the current HTTP Session with server.

Method definition

Class: GlideSession

Definition: HttpSession getSession()

Security impact

N/A

Functional impact

This should not have a functional impact.

Example

```
gs.getSessionToken();
```

5.13 getSession

Gets a reference to the current Glide session.

Example: If you want to retrieve the information tied to current session and perform actions accordingly such as - Restrict any action if current session is initiated by SOAP request and not by regular login.

```
If (gs.getSession().isInteractive()) {allow action} Else {do not allow action}
```

To see if what roles are assigned to current session

```
gs.getSession().getRoles();
```

Method definition

Class: GlideSession

Definition: Session getSession()

Security impact

N/A

Functional impact

This should not have a functional impact.

Example

```
gs.getSession();
```

6. Secure Communication

Secure communication assists in managing the security while the sensitive data is being transferred across the Internet between the client and the server. In this section we will go through various GlideScriptable classes and methods related to secure communication that may be used in customer scripts. We will also illustrate it with ServiceNow examples.

6.1 setSignerKey

SCOPE: This method is NOT accessible in script running in a package scope.

This should be used to set the private key that will be later leveraged to sign the SOAP requests.

When appropriate, call this method during document signing, to set the signer key before signing the document.

Method definition

Class: GlideSOAPDocument

Definition: void setSignerKey(String keyData)

Security impact

Ensure a secure communication channel when transferring sensitive data. Also ensure and validate sender authenticity. ServiceNow allows SOAP Document signatures with a specific key.

Functional impact

This should not have a functional impact.

Example

To set the document signer key, use the following code.

```
GlideSOAPDocument().setSignerKey("key");
```

6.2 setSignerCertificate

SCOPE: This method is NOT accessible in script running in a package scope.

Use this method to set the certificate that will be used to sign the SOAP requests.

Call this method to set the signer certificate before signing the document.

Method definition

Class: GlideSOAPDocument

Definition: void setSignerCertificate(String certData)

Security impact

Ensure a secure communication channel when transferring sensitive data. Also ensure and validate sender authenticity. ServiceNow allows SOAP Document signatures.

Functional impact

This should not have a functional impact.

Example

Use the following code to set the document signer key.

```
GlideSOAPDocument().setSignerCertificate("certificate data");
```

6.3 getSignedDocument

SCOPE: This method is NOT accessible in script running in a package scope.

This method signs the SOAP request and returns a signed document.

Call to this method to sign the document after setting the signer key and certificate.

Method definition

Class: GlideSOAPDocument

Definition: Document getSignedDocument()

Security impact

Ensure a secure communication channel when transferring sensitive data. Also ensure and validate sender authenticity. ServiceNow allows SOAP Document signatures.

Functional impact

This should not have a functional impact.

Example

```
GlideSOAPDocument().getSignedDocument();
```

6.4 wantSigned

SCOPE: This method is NOT accessible in script running in a package scope.

Use this method to check if SOAP request will be signed, and if the instance user will be able to sign the request. It validates if the instance user has set the key and certificate values using above methods.

Call to this method to check if both the key and certificate are set. Returns either True or False.

Method definition

Class: GlideSOAPDocument

Definition: boolean wantSigned()

Security impact

Ensure a secure communication channel when transferring sensitive data. Also ensure sender authenticity. ServiceNow allows SOAP Document signatures.

Functional impact

This should not have a functional impact.

Example

```
GlideSOAPDocument().wantSigned();
```

6.5 setSignatureKeys

SCOPE: This method is NOT accessible in script running in a package scope.

When the private key data and certificate data are set using this method, the SOAP requests are automatically signed.

Use this method to sign all the SOAP requests.

Method definition

Class: GlideSOAPRequest

Definition: void setSignatureKeys(String privateKey, String certificateKey)

Security impact

Ensure a secure communication channel when transferring sensitive data. Also ensure and validate sender authenticity. ServiceNow allows SOAP Document signatures.

Functional impact

This should not have a functional impact.

Example

Use the following code to enforce signing all the SOAP requests.

```
GlideSOAPRequest().setSignatureKeys("private_key","certificate_key");
```

6.6 SOAPSigner

SCOPE: This method is NOT accessible in script running in a package scope.

This is a constructor to create a SOAPSigner instance with a specific key and certificate.

Use the SOAPSigner creates instance to sign documents later.

Method definition

Class: GlideSOAPSigner

Definition: public SOAPSigner(String key, String cert)

Security impact

Ensure a secure communication channel when transferring sensitive data. Also ensure and validate sender authenticity. ServiceNow allows SOAP Document signatures.

Functional impact

This should not have a functional impact.

Example

```
var mySigner= new SOAPSigner("key","certdata")
```

6.7 sign

This method signs a given document and converts it to a SOAP message.

Method definition

Class: GlideSOAPSigner

Definition: String sign(Document d)

Security impact

Ensure a secure communication channel when transferring sensitive data. Also ensure sender authenticity. ServiceNow allows SOAP Document signatures.

Functional impact

This should not have a functional impact.

Example

```
var mySigner = new SOAPSigner("key","certdata")
var mySignedSoap = mySigner.sign(myDocument)
```

7. Secure Data Access

Limit instance data access to only authorized users and roles. Perform strict roles and privileges check before allowing data access. In this section we will discuss GlideScriptable classes and methods related to secure data access that may be used in customer scripts, and illustrate with ServiceNow examples.

7.1 GlideRecord

This is a constructor that allows to instantiate the GlideRecord class and returns an object pointing to the desired database table.

Use this method to access database tables

Method definition

Class: GlideRecord

Definition: public GlideRecord(String tableName)

Security impact

Ensure only users with proper roles are able to access the tables. Use this method with the following ones to check user access privileges before accessing any table data.

- canRead ()
- canDelete()
- canWrite()
- canCreate()

Those methods will return True if the user has enough privileges to access the requested data.

Functional impact

This should not have a functional impact.

Example

Examples and references:

7.2 GlideRecordSecure

GlideRecordSecure is an inherited class that performs the same functions as GlideRecord, and also enforces Security restrictions. Like GlideRecord, GlideRecordSecure is an object that contains zero or more records from one table, or an ordered list that is used for database operations instead of writing SQL queries. GlideRecordSecure supports reference elements and can be used for securing table access via Script includes or any server-side query.

This should be used to access database tables in a secure manner.

Method definition

Class: GlideRecord

Definition: `public GlideRecordSecure(String tableName)`

Security impact

Ensure secure data access, so only users with proper roles are able to access the tables. The GlideRecordSecure checks the privileges of the current user requesting the data against table ACLs before access. With GlideRecordSecure, you do not need to explicitly check for read access using `canRead()`. You may use `next()` by itself to move to the next record. The following table provides a detailed comparison between GlideRecord and GlideRecordSecure.

Functional impact

This should not have a functional impact.

Example

Examples and references:

https://docs.servicenow.com/bundle/helsinki-servicenow-platform/page/script/glide-server-apis/concept/c_UsingGlideRecordSecure.htmlindex.php?title=GlideRecordSecure#gsc.tab=0

8. Logging

Avoid logging any sensitive information such as passwords, encryption keys, etc. One option is to mask the data while logging and using wildcard character (*****) as a placeholder for sensitive information. Do not log sensitive information on the system. Information such as passwords and encryption keys, credit cards numbers etc. belong only to the owner (user) and should not be logged in clear text as those logs may be accessible by other users. This will have a huge potential Security exposure of sensitive information.

8.1 Logging

Ensure no sensitive information such as Passwords, encryption keys, etc. are logged. An option is to mask the data while logging and using wildcard character (*****) as a placeholder for sensitive information.

Method definition

Class: N/A

Definition: N/A()

Security impact

Do not log sensitive information on the system. Information such as passwords and encryption keys, credit cards numbers etc. belong only to the owner (user) and not logged in clear text as those logs may be accessible by other users which will have a critical security impact.

Functional impact

This should not have a functional impact.

Example

N/A

9. Encryption and Hashing

Ensure proper data encryption is performed on sensitive information. In this section we will discuss various GlideScriptable classes and methods that may be used in customer scripts, and review standard guidance for encryption illustrated by ServiceNow examples.

9.1 GlideEncrypter

SCOPE: This method is NOT accessible in script running in a package scope.

This is a constructor intended to create an instance of the GlideEncrypter scriptable class with a specific encryption key. The encryption key should be at least 24 bytes length.

Create an instance of this class with the desired key to be used for various encryption and decryption process. This class is used to encrypt/decrypt strings using TripleDES.

Method definition

Class: GlideEncrypter

Definition: public GlideEncrypter(String encryptionKey)

Security impact

Do not store sensitive information in clear text on the system. Information such as passwords and encryption keys, credit cards numbers etc. belong only to the owner (user) and not stored or transmitted in clear text. Handle such information carefully, with strong encryption mechanisms.

Functional impact

This should not have a functional impact.

Example

Use the following code to instantiate the GlideEncrypter class with the key "123456789ABCDEFGHJKLMNOP".

```
var encrypter=new GlideEncrypter("123456789ABCDEFGHJKLMNOP ")
```

9.2 encrypt

SCOPE: This method is NOT accessible in script running in a package scope.

This method encrypts a string using the passphrase supplied along with the constructor and returns a base64 encoded representation of the encrypted string

This method is used to encrypt strings using TripleDES.

Method definition

Class: GlideEncrypter

Definition: String encrypt(String unencryptedString)

Security impact

Do not store sensitive information in clear text on the system. Information such as passwords and encryption keys, credit cards numbers etc. belong only to the owner (user) and not stored or transmitted in clear text. Handle such information carefully, with strong encryption mechanisms.

Functional impact

This should not have a functional impact.

Example

```
var myClearText="Hello world"  
var encrypter = new GlideEncrypter("123456789ABCDEFGHJKLMNO")  
var encryptedStringAsBase64 = encrypter.encrypt(myClearText)  
gs.print(encryptedStringAsBase64)
```

Result:

```
*** Script: V+PP3lDJic0KMMEKpaUWBg==
```

9.3 decrypt

SCOPE: This method is NOT accessible in script running in a package scope.

This method decrypts a base64 encoded cypher using the passphrase supplied along with the constructor and returns the original clear text string.

This method is used to decrypt cyphers using TripleDES.

Method definition

Class: GlideEncrypter

Definition: String decrypt(String encryptedString)

Security impact

Do not store sensitive information in clear text on the system. Information such as passwords and encryption keys, credit cards numbers etc. belong only to the owner (user) and not stored or transmitted in clear text. Handle such information carefully, with strong encryption mechanisms.

Functional impact

This should not have a functional impact.

Example

```
var cypher="V+PP3lDJic0KMMEKpaUWBg=="  
var encrypter = new GlideEncrypter("123456789ABCDEFGHJKLMNO")  
var decryptedString = encrypter.decrypt(cypher)  
gs.print(decryptedString)
```

Result:

```
*** Script: Hello world
```

9.4 generateKey

SCOPE: This method is NOT accessible in script running in a package scope.

Use this method to generate a random (encoded) encryption key when user does not wish to provide one himself/herself. The bytes for the key are returned as a hex string.

Method definition

Class: GlideEncryptionContextCipher

Definition: String generateKey(int sizeBytes)

Security impact

Do not store sensitive information in clear text on the system. Information such as passwords and encryption keys, credit cards numbers etc. belong only to the owner (user) and not stored or transmitted in clear text. Handle such information carefully, with strong encryption mechanisms.

Functional impact

This should not have a functional impact.

Example

```
var MyContext = new GlideEncryptionContextCipher()  
var myHexRandomKey = MyContext.generateKey(32)
```

9.5 generate

This method may be used to generate a random GUID. It returns a 32 character GUID in the form of a 32 character hex value (representing 128 binary bits).

Method definition

Class: GlideGuid

Definition: String generate(Object obj)

Security impact

This method does not use SecureRandom number generator. Do not use the generated GUID for critical process such as authentication, unique session identifier etc.

Use a secureRandom number generator for critical processes or encryption, to ensure a truly random number.

Functional impact

This should not have a functional impact.

Example

```
var myGuidInstance= new GlideGuid()  
var myGuid= myGuidInstance.generate(null)  
gs.print(myGuid)
```

Result:

```
*** Script: 6914e43adb93b20017df78b5ae96190a
```

9.6 generateRand

SCOPE: This method is NOT accessible in script running in a package scope.

This method is similar to "generate" and may be used to generate a random GUID. It returns a 32 character GUID in the form of a 32 character hex value (representing 128 binary bits).

Method definition

Class: GlideGuid

Definition: String generateRand ()

Security impact

This method does not use SecureRandom number generator. Do not use the generated GUID for critical process such as authentication, unique session identifier etc.

Use a secureRandom number generator for critical processes or encryption, to ensure a truly random number.

Functional impact

This should not have a functional impact.

Example

```
var myGuidInstance = new GlideGuid()  
var myGuid = myGuidInstance.generateRand()  
gs.print(myGuid)
```

Result:

```
*** Script: 8264643a2193b2002c641e5f03b7e6d7
```

9.7 generateSecureRand

SCOPE: This method is NOT accessible in script running in a package scope.

This method uses a secure random number generator to generate a unique random GUID, it returns a 32 character GUID in the form of a 32 character hex value (representing 128 binary bits).

Method definition

Class: GlideGuid

Definition: String generateSecureRand ()

Security impact

This method uses a SecureRandom number generator. Use the generated GUID for critical process such as authentication, unique session identifier etc.

Use a secureRandom number generator for critical processes or encryption, to ensure a truly random number.

Functional impact

This should not have a functional impact.

Example

```
var myGuidInstance = new GlideGuid()  
var myGuid = myGuidInstance.generateSecureRand()  
gs.print(myGuid)
```

Result:

```
*** Script: 4155e43a7e93b200a737e697da7b710c
```

9.8 getRandomString

SCOPE: This method is NOT accessible in script running in a package scope.

Use this method to generate a random string with specific length. The Method will return a random string with the following character set.

- a - z
- A - Z
- 0 - 9

Method definition

Class: GlideGuid

Definition: String getRandomString(int length)

Security impact

This method does not use a SecureRandom number generator. Do not use the generated string for the critical processes such as authentication, unique session identifier etc.

Use a secureRandom number generator for critical processes or encryption, to ensure a truly random number.

Functional impact

This should not have a functional impact.

Example

```
var myGenerator = new GlideStringUtil()  
var myRandomString = myGenerator.getRandomString(32)  
gs.print(myRandomString)
```

Result:

```
*** Script: 7KXhh1SndX8YQsX8a016i45xmwnv3DYe
```

The world works with ServiceNow.

Was this content helpful?

 Yes

 No

74% found this useful

Did this KB article help you?



servicenow

The world works with ServiceNow.

[Terms and conditions](#)

[Privacy statement](#)

[GDPR](#)

[Cookie policy](#)

[Cookie Preferences](#)



© 2024 ServiceNow. All rights reserved.